# 重要知识点100条

1、企业集成分类：按传输方式分

|  |  |
| --- | --- |
|  | 特点 |
| 消息集成 | 数据量小，交互频繁，立即地，异步 |
| 共享数据库 | 交互频繁，立即地，同步 |
| 文件传输 | 数据量大，交互频度小，即时性要求低（月末，年末） |

2、企业集成分类：按集成点分

|  |  |  |  |
| --- | --- | --- | --- |
|  | 集成点 | 效果 | 解题关键点 |
| 界面集成 | 界面 | 统一入口，产生“整体”感觉 | “整体”感觉  最小代价实现一体化操作 |
| 数据集成 | 数据 | 不同来源的数据逻辑或物理上“集中” | 其他集成方法的基础 |
| 控制集成 | 应用逻辑 | 调用其他系统已有方法，达到集成效果 |  |
| 业务流程集成（过程集成） | 应用逻辑 | 跨企业，或优化流程而非直接调用 | 企业之间的信息共享能力 |
| 门户集成 |  | 将内部系统对接到互联网上 | 发布到互联网上 |

3、系统工程方法

|  |  |
| --- | --- |
| 系统工程方法 | 关键点 |
| 霍尔三维结构  “硬科学”方法论 | 逻辑维：逻辑维即解决问题的逻辑过程。  时间维：时间维即是工作进程。  知识维：知识维即是专业科学知识。  应用场景：组织和管理大型工程建设项目。 |
| 切克兰德方法  “软科学”方法论 | 核心不是“最优化”，而是“比较”和“探寻”。  7步骤：认识问题、根底定义、建立概念模型、比较及探寻、选择、设计与实施、评估与反馈 |
| 并行工程方法 | “制造过程”与“支持过程”并行。  强调三个方面：产品设计开发期间，最快速度按质完成；各项工作问题协调解决；适当的信息系统工具。 |
| 综合集成法 | 钱学森命名，【简单系统】和【巨系统】  四原则：整体论原则、相互联系原则、有序性原则、动态原则 |
| WSR系统方法 | 实践准则：【懂物理】-【明事理】-【通人理】 |

4、信息系统战略规划

信息系统战略规划（Information System Strategic Planning，ISSP）是从企业战略出发，构建企业基本的信息架构，对企业内、外信息资源进行统一规划、管理与应用，利用信息控制企业行为，辅助企业进行决策，帮助企业实现战略目标。

ISSP方法经历了三个主要阶段，各个阶段所使用的方法也不一样。

第一个阶段主要以数据处理为核心，围绕职能部门需求的信息系统规划，主要的方法包括企业系统规划法（BSP）--CU矩阵、关键成功因素法（CSF）和战略集合转化法（SST）；

第二个阶段主要以企业内部管理信息系统为核心，围绕企业整体需求进行的信息系统规划，主要的方法包括战略数据规划法（SDP）、信息工程法（IE）和战略栅格法（SG）；

第三个阶段的方法在综合考虑企业内外环境的情况下，以集成为核心，围绕企业战略需求进行的信息系统规划，主要的方法包括价值链分析法（VCA）和战略一致性模型（SAM）。

5、电子政务

政府对政府（G2G，Government To Government）：政府之间的互动及政府与公务员之间互动。基础信息的采集、处理和利用，如人口/地理/资源信息等；各级政府决策支持；政府间通信。

政府对企业（G2B，Government To Business）：政府为企业提供的政策环境。包括产业政策、进出口、注册、纳税、工资、劳保、社保等各种规定；政府向企事业单位颁发的各种营业执照、许可证、合格证、质量认证等。

企业对政府（B2G，Business To Government）：企业纳税及企业为政府提供服务。企业参加政府各项工程的竞/投标，向政府供应各种商品和服务，企业向政府提建议，申诉。

政府对公民（G2C，Government To Citizen）：政府对公民提供的服务。关于社区公安和水、火、天灾等与公共安全有关的信息等，还包括户口、各种证件的管理等政府提供的各种服务。

公民对政府（C2G，Citizen To Government）：个人应向政府缴纳税费和罚款及公民反馈渠道。个人应向政府缴纳的各种税款和费用。了解民意，征求群众意见。报警服务（盗贼、医疗、急救、火警等）。

政府对公务员（G2E，Government To Employee）：政府与政府公务员即政府雇员。政府机构通过网络技术实现内部电子化管理（例如，OA系统等）的重要形式。

6、与逆向工程相关的概念有重构、设计恢复、再工程和正向工程。

（1）重构（restructuring）。重构是指在同一抽象级别上转换系统描述形式。

（2）设计恢复（design recovery）。设计恢复是指借助工具从已有程序中抽象出有关数据设计、总体结构设计和过程设计等方面的信息。

（3）逆向工程（reverse engineering）：逆向工程是分析程序，力图在比源代码更高抽象层次上建立程序的表示过程，逆向工程是设计的恢复过程。

（4）正向工程（forward engineering）。正向工程是指不仅从现有系统中恢复设计信息，而且使用该信息去改变或重构现有系统，以改善其整体质量。

（5）再工程（re-engineering）。再工程是对现有系统的重新开发过程，包括逆向工程、新需求的考虑过程和正向工程三个步骤。

7、结构化开发方法

用户至上，自顶向下，逐步分解（求解），严格区分工作阶段，每阶段有任务与成果，强调系统开发过程的整体性和全局性，系统开发过程工程化，文档资料标准化。阶段固化，不善变化，适用于需求明确。

8、原型法开发方法

适用于需求不明确的开发，按功能分-水平原型（界面）、垂直原型（复杂算法），按最终结果分为抛弃式原型和演化式原型。

9、面向对象方法

最早来源于仿真领域，其特点是系统的描述及信息模型的表示与客观实体相对应，符合人们的思维习惯，有利于系统开发过程中用户与开发人员的交流和沟通，缩短开发周期，提供系统开发的准确性和效率。具有更好的复用性，关键在于建立一个全面、合理、统一的模型，分析、设计、实现三个阶段界限不明确。

10、面向服务的方法

以粗粒度、松散耦合的系统功能为核心，强调系统功能的标准化和构件化，加强了系统的灵活性、可复用性和可演化性。

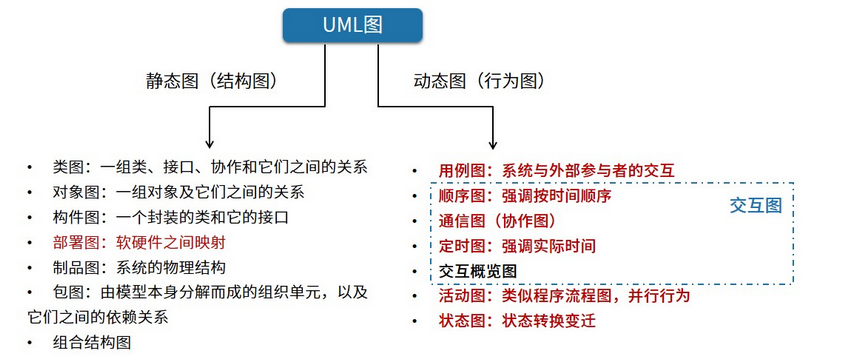
从概念上讲，SOA 中有三个主要的抽象级别：

操作：代表单个逻辑工作单元（LUW）的事务。执行操作通常会导致读、写或修改一个或多个持久性数据。SOA 操作可以直接与面向对象 (OO) 的方法相比。它们都有特定的结构化接口，并且返回结构化的响应。完全同方法一样，特定操作的执行可能涉及调用附加的操作。操作位于最底层。

服务：代表操作的逻辑分组。例如，如果我们将 CustomerProfiling视为服务，则按照电话号码查找客户、 按照名称和邮政编码列出顾客和 保存新客户的数据就代表相关的操作。

业务流程：为实现特定业务目标而执行的一组长期运行的动作或活动。业务流程通常包括多个业务调用。业务流程的例子有： 接纳新员工、 出售产品或服务和完成订单。

11、UML图分类



12、面向对象设计原则

单一职责原则：设计目的单一的类

开放-封闭原则：对扩展开放，对修改封闭

李氏(Liskov)替换原则：子类可以替换父类

依赖倒置原则：要依赖于抽象，而不是具体实现；针对接口编程，不要针对实现编程

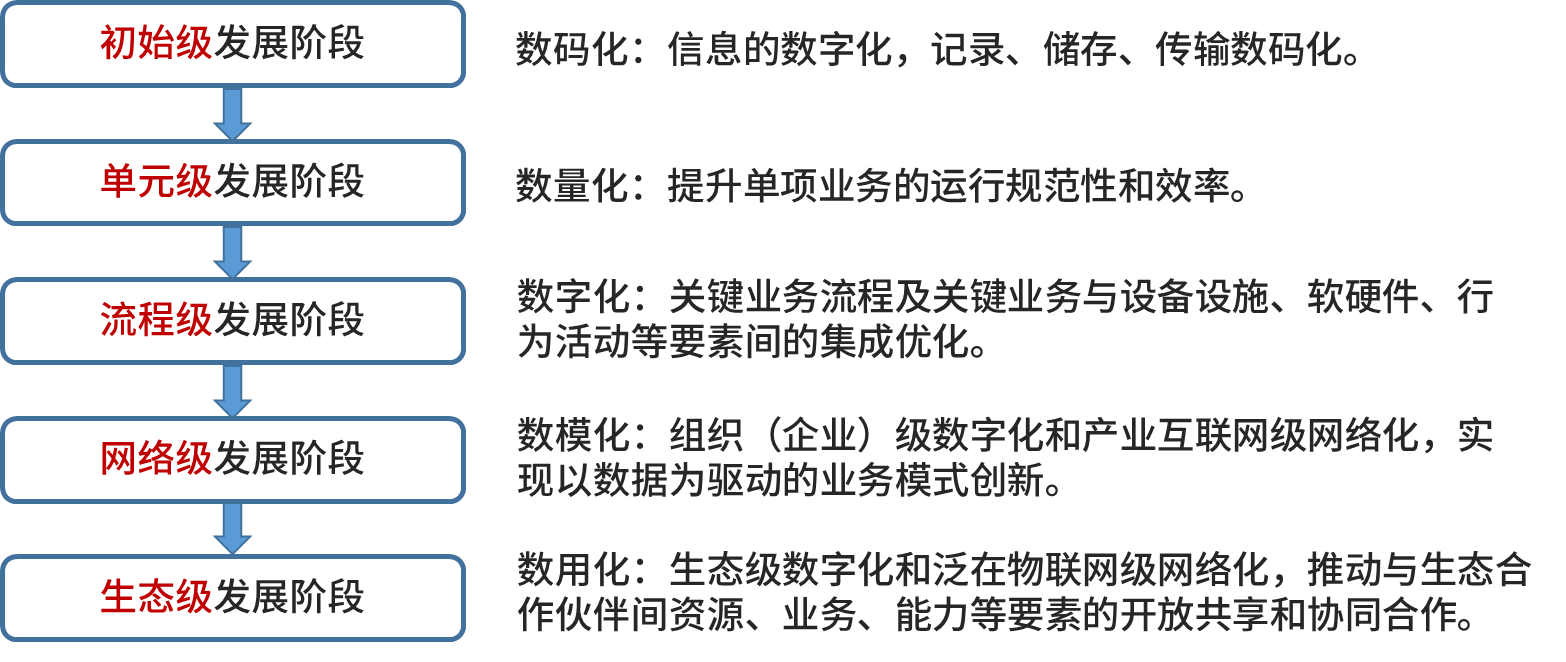
接口隔离原则：使用多个专门的接口比使用单一的总接口要好

组合重用原则：要尽量使用组合，而不是继承关系达到重用目的

迪米特(Demeter)原则(最少知识法则)：一个对象应当对其他对象有尽可能少的了解

13、数字化转型的5个发展阶段

企业数字化转型的五个发展阶段分别为：初始级发展阶段、单元级发展阶段、流程级发展阶段、网络级发展阶段、生态级发展阶段。



14、信息系统的分类

|  |  |
| --- | --- |
| 信息系统的分类 | 关键点 |
| 业务处理系统  【TPS】 | 早期最初级的信息系统【20世纪50-60年代】。  功能：数据输入、数据处理【批处理、OLTP】、数据库维护、文件报表产生。 |
| 管理信息系统  【MIS】 | 高度集成化的人机信息系统。  金字塔结构：分多个层级。 |
| 决策支持系统  【DSS】 | 由语言系统、知识系统和问题处理系统组成。  用于辅助决策、支持决策。 |
| 专家系统【ES】 | 知识 + 推理 = 专家系统。人工智能的一个重要分支。 |
| 办公自动化系统  【OAS】 | 由计算机设备、办公设备、数据通信及网络设备、软件系统组成。 |
| 企业资源计划  【ERP】 | 打通供应链，集成，整合。 |

15、系统工程生命周期阶段及周期方法

阶段：探索性研究→使用阶段→概念阶段→开发阶段→生产阶段→保障阶段→退役阶段

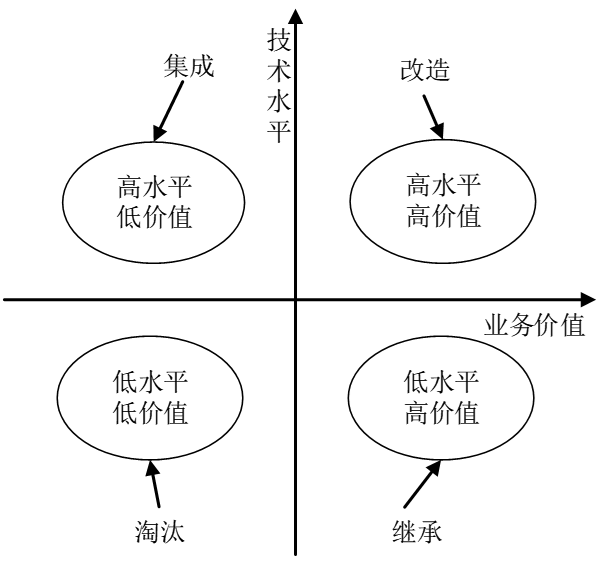
方法：计划驱动方法：需求→设计→构建→测试→部署

渐进迭代式开发：提供连续交付以达到期望的系统

精益开发：起源于丰田，是一个动态的、知识驱动的，以客户为中心的过程

敏捷开发：更好的灵活性

16、遗留系统演化策略



淘汰策略：遗留系统的技术含量较低，且具有较低的业务价值。对遗留系统的完全淘汰是企业资源的根本浪费，系统分析师应该善于“变废为宝”，通过对遗留系统功能的理解和借鉴，可以帮助新系统的设计，降低新系统开发的风险。

继承策略：遗留系统的技术含量较低，已经满足企业运作的功能或性能要求，但具有较高的商业价值，目前企业的业务尚紧密依赖该系统。对这种遗留系统的演化策略为继承。在开发新系统时，需要完全兼容遗留系统的功能模型和数据模型。为了保证业务的连续性，新老系统必须并行运行一段时间，再逐渐切换到新系统上运行。

改造策略：遗留系统具有较高的业务价值，基本上能够满足企业业务运作和决策支持的需要。这种系统可能建成的时间还很短，对这种遗留系统的演化策略为改造。改造包括系统功能的增强和数据模型的改造两个方面。系统功能的增强是指在原有系统的基础上增加新的应用要求，对遗留系统本身不做改变；数据模型的改造是指将遗留系统的旧的数据模型向新的数据模型的转化。

集成策略：遗留系统的技术含量较高，但其业务价值较低，可能只完成某个部门（或子公司）的业务管理。这种系统在各自的局部领域里工作良好，但对于整个企业来说，存在多个这样的系统，不同的系统基于不同的平台、不同的数据模型，形成了一个个信息孤岛，对这种遗留系统的演化策略为集成。

17、新旧系统转换策略

直接转换：接转换是在原有系统停止运行的某一时刻，新系统立即投入运行，中间没有过渡阶段。采用这种方式时，人力和费用最省，适用于系统不太复杂或现有系统完全不能使用的场合。但是这种方式风险高。

并行转换：并行转换就是新系统和旧系统并行工作一段时间，经过这段时间的试运行后，再用新系统正式替换下现有系统。那么这种方式，它的好处就是风险很小。在转换期间还可以同时比较新旧两套系统的性能，而且能够让操作人员得到全面的培训，所以对于一些比较大的信息系统，或者处理过程比较复杂，数据比较重要的系统。并行转换是一种最常用的转换方式。那么这种转换方式也有缺点，缺点就在于两套系统并行期间，要有两套班子或者两套处理方式同时并存，在人力和费用消耗比较大，转换的周期比较长，而且难以控制新旧系统当中数据的变化。所以这就要求要做好转换计划，并且要加强管理。

分段转换：这是直接转换和并行转换的接合，也就是分期分批、逐步转换。一般比较大的系统采用这种方式比较合适，他能够保证软件平稳运行，费用也不太高，就是将大的系统分成多个子系统，每成熟一个子系统就切换一个子系统，主要是分期分批。这种分段转换的策略，它的优点就是成熟一个子系统就转换一个子系统。这种新旧转换，震动比较小，用户比较容易接受。但是由于采取的是渐进的方式，会导致新旧系统的转换周期比较长。

18、系统维护分类

正确性维护：指改正在系统开发阶段已发生而系统测试阶段尚未发现的错误。

适应性维护：指使应用软件适应环境变化【外部环境、数据环境】而进行的修改。

完善性维护：扩充功能和改善性能而进行的修改。

预防性维护：为了适应未来的软硬件环境的变化，应主动增加预防性的新的功能，以使用系统适应各类变化而不被淘汰。如将专用报表功能改成通用报表生成功能，以适应将来报表格式的变化。

19、可行性分类

经济可行性：成本收益分析，包括建设成本、运行成本和项目建设后可能的经济收益。

技术可行性：技术风险分析，现有的技术能否支持系统目标的实现，现有资源（员工，技术积累，构件库，软硬件条件）是否足以支持项目的实施。

法律可行性(社会可行性)：不能与国家法律或政策相抵触。

用户使用可行性：执行可行性，从信息系统用户的角度评估系统的可行性。

管理可行性：系统与现有管理机制的一致性，改革的可能性。

运行可行性：用户方便使用的程度。

20、基于构件的软件工程（CBSE）

CBSE体现了“购买而不是重新构造”的哲学。

CBSE的构件应该具备的特征：

1、可组装性：所有外部交互必须通过公开定义的接口进行。

2、可部署性：构件总是二进制形式的，能作为一个独立实体在平台上运行。

3、文档化：用户根据文档来判断构件是否满足需求。

4、独立性：可以在无其他特殊构件的情况下进行组装和部署。

5、标准化：符合某种标准化的构件模型。

【构件的组装】：

1、顺序组装：按顺序调用已经存在的构件，可以用两个已经存在的构件来创造一个新的构件。

2、层次组装：被调用构件的“提供”接口必须和调用构件的“请求”接口兼容。

3、叠加组装：多个构件合并形成新构件，新构件整合原构件的功能，对外提供新的接口。

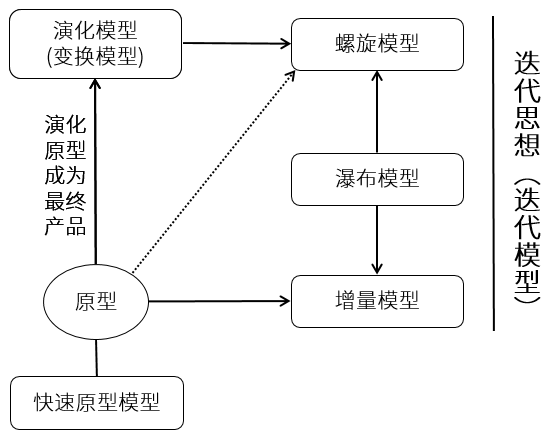
21、信息安全包括5个基本要素

|  |  |
| --- | --- |
| 名称 | 说明 |
| 机密性 | 机密性是指网络信息不泄露给非授权的用户、实体或程序，能够防止非授权者获取信息。 |
| 完整性 | 完整性是指网络信息或系统未经授权不能进行更改的特性。 |
| 可用性 | 可用性是指合法许可的用户能够及时获取网络信息或服务的特性。 |
| 可控性 | 可控性是指可以控制授权范围内的信息流向及行为方式。 |
| 可审查性 | 可审查性是指对出现的信息安全问题提供调查的依据和手段。 |

22、原型模型

典型的原型开发方法模型。适用于需求不明确的场景，可以帮助用户明确需求。

原型的发展方向:



23、瀑布模型

瀑布模型是将软件生存周期中的各个活动规定为依线性顺序连接的若干阶段的模型，包括需求分析、设计、编码、运行与维护。

瀑布模型的特点是容易理解，管理成本低，每个阶段都有对应的成果产物，各个阶段有明显的界限划分和顺序要求，一旦发生错误，整个项目推倒重新开始。

适用于需求明确的项目，一般表述为需求明确或二次开发，或者适用于需要对数据处理的项目

24、增量模型

融合了瀑布模型的基本成分和原型实现的迭代特征，可以有多个可用版本的发布，核心功能往往最先完成，在此基础上，每轮迭代会有新的增量发布，核心功能可以得到充分测试。强调每一个增量均发布一个可操作的产品。

25、螺旋模型

典型特点是引入了风险分析。结合了瀑布模型和演化模型的优点，最主要的特点在于加入了风险分析。它是由制定计划、风险分析、实施工程、客户评估这一循环组成的，并且从概念项目开始第一个螺旋。

26、云原生架构风格

（1）云计算基本概念：

云计算是集合了大量计算设备和资源，对用户屏蔽底层差异的分布式处理架构，其用户与提供实际服务的计算资源是相分离的。

云计算优点：超大规模、虚拟化、高可靠性、高可伸缩性、按需服务、成本低【前期投入低、综合使用成本也低】。

（2）分类

按服务类型分类：

|  |  |
| --- | --- |
| SaaS【软件即服务】 | 基于多租户技术实现，直接提供应用程序 |
| PaaS【平台即服务】 | 虚拟中间件服务器、运行环境和操作系统 |
| IaaS【基础设施即服务】 | 包括服务器、存储和网络等服务 |

按部署方式分类：

公有云：面向互联网用户需求，通过开放网络提供云计算服务

私有云：面向企业内部提供云计算服务

混合云：兼顾以上两种情况的云计算服务

27、敏捷开发

敏捷开发是一种以人为核心、迭代、循序渐进的开发方法，适用于小团队和小项目，具有小步快跑的思想。常见的敏捷开发方法有极限编程法、水晶法、并列争球法和自适应软件开发方法。

28、统一过程（在软考中UP、RUP都指统一过程）

典型特点是用例驱动、以架构为中心、迭代和增量。

统一过程把一个项目分为四个不同的阶段：

构思阶段（初始/初启阶段）：定义最终产品视图和业务模型、确定系统范围。

细化阶段（精化阶段）：设计及确定系统架构、制定工作计划及资源要求。

构造阶段：开发剩余构件和应用程序功能，把这些构件集成为产品，并进行详细测试。

移交阶段：确保软件对最终用户是可用的，进行β测试，制作产品发布版本。

9个核心工作流：业务建模、需求、分析与设计、实现、测试、部署、配置与变更管理、项目管理、环境。

32、基于服务的架构（SOA）

服务是一种为了满足某项业务需求的操作、规则等的逻辑组合，它包含一系列有序活动的交互，为实现用户目标提供支持。

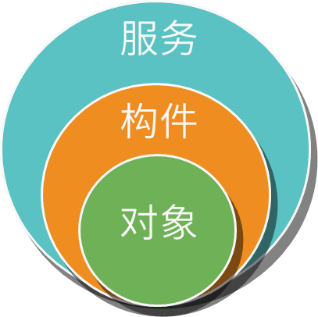
（1）典型的SOA架构



（2）单个服务的内部结构



（3）SOA层次和特点



服务是标准化程度更高的构件。

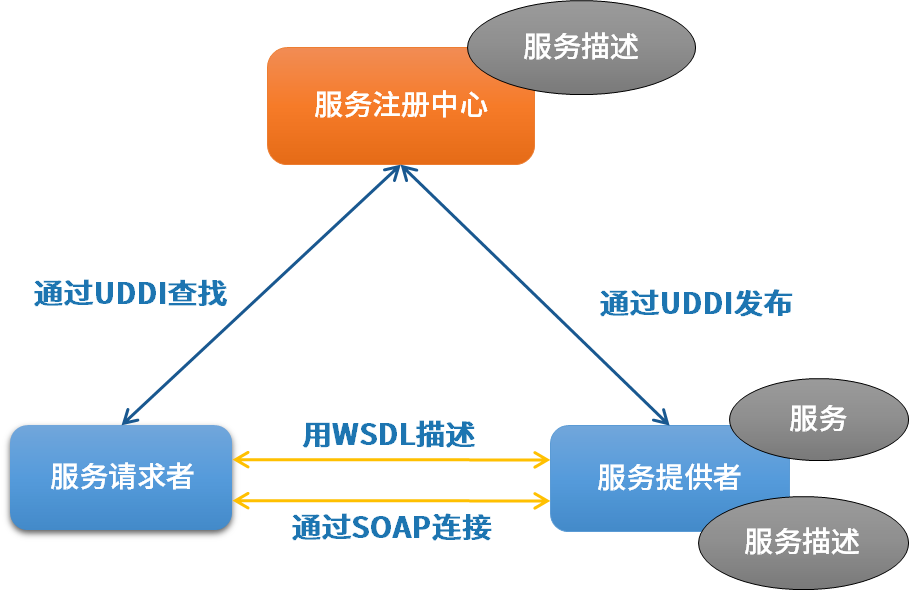
服务构件粗粒度，传统构件细粒度居多（粗粒度）。

服务构件的接口是标准的，主要是WSDL接口，传统构件常以具体API形式出现（标准化结构）。

服务构件的实现与语言无关，传统构件绑定某种特定语言。

服务构件可以通过构件容器提供QoS的服务，传统构件完全由程序代码直接控制（松耦合）。

（4）关键技术



|  |  |
| --- | --- |
| 功能 | 协议 |
| 发现服务 | UDDI、DISCO |
| 描述服务 | WSDL、XML Schema |
| 消息格式层 | SOAP、REST |
| 编码格式层 | XML（DOM，SAX） |
| 传输协议层 | HTTP、TCP/IP 、SMTP等 |

33、微服务

（1）什么是微服务

微服务顾名思义，就是很小的服务，所以它属于面向服务架构的一种。

（2）微服务的优势

|  |  |
| --- | --- |
| 优点 | 解读 |
| 【复杂应用解耦】 | 小服务（且专注于做⼀件事情）  化整为零，易于小团队开发 |
| 【独立】 | 独立开发  独立测试及独立部署（简单部署）  独立运行（每个服务运行在其独立进程中） |
| 【技术选型灵活】 | 支持异构（如：每个服务使用不同数据库） |
| 【容错】 | 故障被隔离在单个服务中，通过重试、平稳退化等机制实现应用层容错 |
| 【松耦合，易扩展】 | 可根据需求独立扩展 |

（3）微服务面临的挑战

分布式环境下的数据一致性【更复杂】

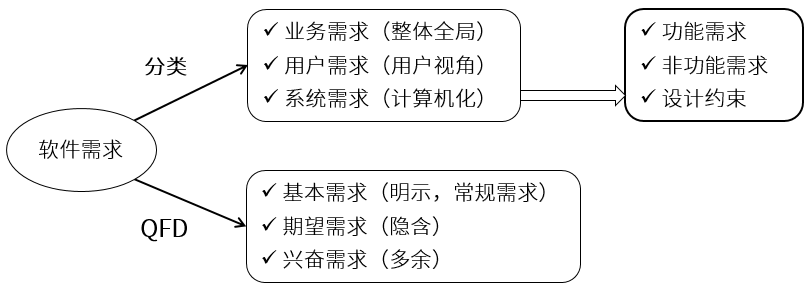
测试的复杂性【服务间依赖测试】

运维的复杂性

29、微服务与SOA对比

|  |  |
| --- | --- |
| 微服务 | SOA |
| 能拆分的就拆分 | 是整体的，服务能放一起的都放一起 |
| 纵向业务划分 | 是水平分多层 |
| 由单一组织负责 | 按层级划分不同部门的组织负责 |
| 细粒度 | 粗粒度 |
| 两句话可以解释明白 | 几百字只相当于SOA的目录 |
| 独立的子公司 | 类似大公司里面划分了一些业务单元（BU） |
| 组件小 | 存在较复杂的组件 |
| 业务逻辑存在于每一个服务中 | 业务逻辑横跨多个业务领域 |
| 使用轻量级的通信方式，如HTTP | 企业服务总线（ESB）充当了服务之间通信的角色 |

30、需求分类



（1）需求的层次

业务需求：是指反应企业或客户对系统高层次的目标要求，通常来自项目投资人、购买产品的客户、客户单位的管理人员、市场营销部门或产品策划部门等。通过业务需求可以确定项目视图和范围，为以后的开发工作奠定了基础。

用户需求：描述的是用户的具体目标，或用户要求系统必须能完成的任务。也就是说，用户需求描述了用户能使用系统来做些什么。

系统需求：是从系统的角度来说明软件的需求，包括功能需求、非功能需求和设计约束等。功能需求也称为行为需求，它规定了开发人员必须在系统中实现的软件功能，用户利用这些功能来完成任务，满足业务要求。非功能需求是指系统必须具备的属性或品质，又可细分为软件质量属性和其他非功能需求。设计约束也称为限制条件或补充规约，通常是对系统的一些约束说明。

（2）QFD

质量功能部署QFD是一种将用户要求转化成软件需求的技术，其目的是最大限度地提升软件工程过程中用户的满意度。QFD将软件需求分为三类：

常规需求（基本需求）：用户认为系统应该做到的功能或性能，实现越多用户会越满意。

期望需求：用户想当然认为系统应具备的功能或性能，但并不能正确描述自己想要得到的这些功能或性能需求。如果期望需求没有得到实现，会让用户感到不满意。

兴奋需求（意外需求）：是用户要求范围外的功能或性能，实现这些需求用户会更高兴，但不实现也不影响其购买的决策。

31、需求获取方法

|  |  |
| --- | --- |
| 方法 | 特点 |
| 收集资料 | 把与系统有关的、对系统开发有益的信息收集起来。 |
| 用户访谈 | 1对1-3，有代表性的用户。成本高。 |
| 问卷调查 | 用户多，无法一一访谈。成本低。 |
| 现场观摩 | 针对较为复杂的流程和操作。 |
| 参加业务实践 | 有效地发现问题的本质和寻找解决问题的办法。 |
| 联合需求计划（JRP） | 高度组织的群体会议，各方参与，成本高。 |
| 阅读历史文档 | 对收集数据性的信息较为有用。 |
| 情节串联板 | 一系列图片，通过这些图片来讲故事。 |
| 抽样调查 | 降低成本。 |

34、企业信息化涉及三类创新

【技术创新】在生产工艺设计、产品设计中使用计算机辅助设计系统，并通过互联网及时了解和掌握创新的技术信息，加快从技术向生产的转化。还有，生产技术与信息技术相结合，能够大幅度地提高技术水平和产品的竞争力。

【管理创新】按照市场发展的要求，要对企业现有的管理流程重新整合，从作为管理核心的财务、资金管理，转向技术、物资、人力资源的管理，并延伸到企业技术创新、工艺设计、产品设计、生产制造过程的管理，进而扩展到客户关系管理、供应链的管理乃至发展到电子商务。

【制度创新】那些不适应企业信息化的管理体制、管理机制和管理制度必须得到创新。

35、主动冗余与被动冗余

主动冗余是指，处理结点出故障之后，结点本身或控制器会协调没有出故障的结点提供处理服务。

被动冗余，则是靠请求者自己完成这个工作，请求者如果发现一台服务器不能正常提供服务，转而找另一台服务器。

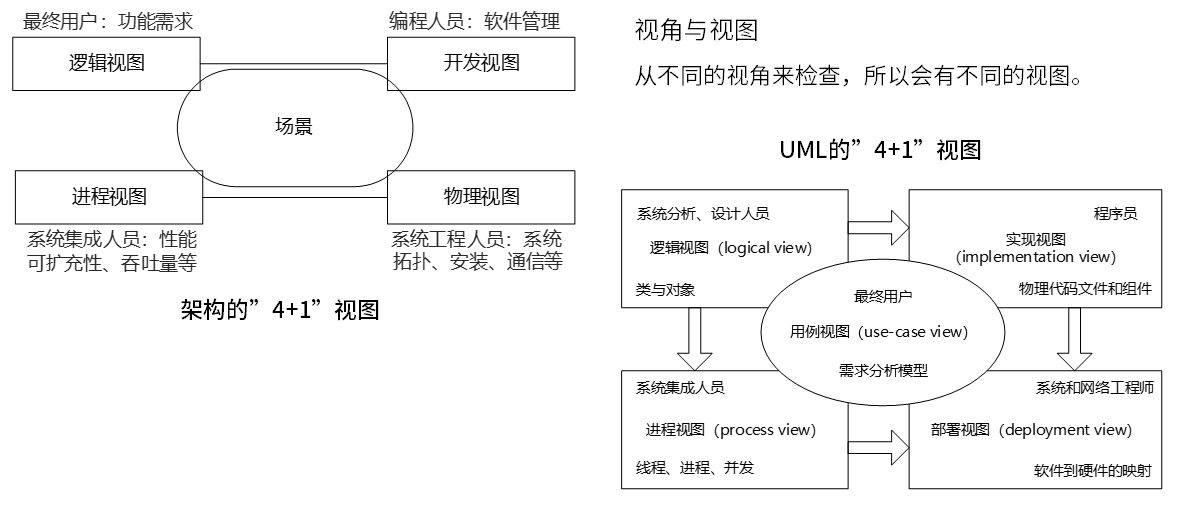
36、人机界面设计—“黄金三准则”

（1）置于用户控制之下

（2）减少用户的记忆负担

（3）保持界面的一致性

37、架构“4+1”视图



结构模型：以架构的构件、连接件和其他概念来刻画结构

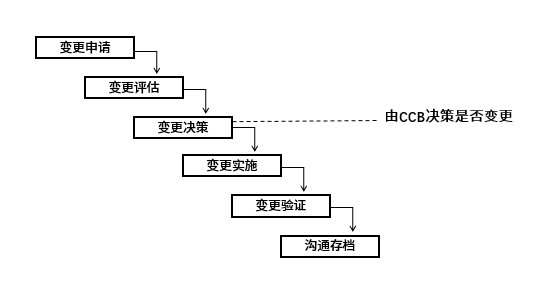
框架模型：不太侧重描述结构的细节而更侧重于整体的结构

动态模型：系统的“大颗粒”的行为性质

过程模型：构建系统的步骤和过程

功能模型：由一组功能构件按层次组成，下层向上层提供服务

38、需求变更管理过程



一般来说，变更控制应该遵循以下的基本流程：

（1）变更申请。应记录变更的提出人、日期、申请变更的内容等信息。

（2）变更评估。对变更的影响范围、严重程度、经济和技术可行性进行系统分析。

（3）变更决策。由CCB决定是否实施变更。

（4）变更实施。由管理者指定的工作人员在受控状态下实施变更。

（5）变更验证。由配置管理人员或受到变更影响的人对变更结果进行评价，确定变更结果和预期是否相符、相关内容是否进行了更新、工作产物是否符合版本管理的要求。

（6）沟通存档。将变更后的内容通知可能会受到影响的人员，并将变更记录汇总归档。如提出的变更在决策时被否决，其初始记录也应予以保存。

39、软件架构风格

架构风格定义了用于描述系统的术语表和一组指导构建系统的规则

|  |  |
| --- | --- |
| 五大架构风格 | 子风格 |
| 数据流风格  【Data Flow】 | 批处理【Batch Sequential】、管道-过滤器【Pipes and Filters】 |
| 调用/返回风  【Call/Return】 | 主程序/子程序【Main Program and Subroutine】  面向对象【Object-oriented】、分层架构【Layered System】 |
| 独立构件风格  【Independent  Components】 | 进程通信【Communicating Processes】、  事件驱动系统（隐式调用）【Event system】 |
| 虚拟机风格  【Virtual Machine】 | 解释器【interpreter】、规则系统【Rule-based System】 |
| 以数据为中心  【Data-centered】 | 数据库系统【Database System】、黑板系统【Blackboard System】、超文本系统【Hypertext System】 |

40、中台

概念：中台是一套结合互联网技术和行业特性，将企业核心能力以共享服务形式沉淀，形成“大中台、小前台”的组织和业务机制，供企业快速低成本的进行业务创新的企业架构。中台又可以进一步细分，比如业务中台、数据中台、XX中台。本质上，都是对企业通用能力在不同层面的沉淀，并对外能力开放。

实例：

Supercell：芬兰移动游戏巨头，2015年世界游戏前10占5席，员工仅200多人，因使用中台，具有小团队快速开发能力，后被腾讯86亿美金收购。

阿里：2015年参观Supercell，而后推行中台。

业务中台&数据中台：

多个电商渠道使用一个下单服务，一个订单接口同时为多个前台系统提供服务，这是业务中台提供的能力。

多个前台系统，根据一个用户的手机号，获取对应的画像，用户的标签，这是数据中台提供的服务。

将多个支付通道，抽象建立成一个支付API，暴露给前台业务系统，这是业务中台提供的能力。

通过一个订单编号，来获取可能的商品推荐清单，从而做到交叉销售，这是数据中台提供的服务。

41、数据中台必备的4个核心能力：

（1）数据汇聚整合能力

（2）数据提纯加工能力

（3）数据服务可视化

（4）价值变现方面

42、模块独立性的度量——聚合：衡量模块内部各元素结合的紧密程度

偶然聚合：模块完成的动作之间没有任何关系，或者仅仅是一种非常松散的关系。

逻辑聚合：模块内部的各个组成在逻辑上具有相似的处理动作，但功能用途上彼此无关。

时间聚合：模块内部的各个组成部分所包含的处理动作必须在同一时间内执行。

过程聚合：模块内部各个组成部分所要完成的动作虽然没有关系，但必须按特定的次序执行。

通信聚合：模块的各个组成部分所完成的动作都使用了同一个数据或产生同一输出数据。

顺序聚合：模块内部的各个部分，前一部分处理动作的最后输出是后一部分处理动作的输入。

功能聚合：模块内部各个部分全部属于一个整体，并执行同一功能，且各部分对实现该功能都必不可少

43、模块独立性的度量——耦合：度量不同模块间互相依赖的程度

非直接耦合：两个模块之间没有直接关系，它们的联系完全是通过主模块的控制和调用来实现的。

数据耦合：两个模块彼此间通过数据参数交换信息。

标记耦合：一组模块通过参数表传递记录信息，这个记录是某一个数据结构的子结构，而不是简单变量。

控制耦合：两个模块彼此间传递的信息中有控制信息。

外部耦合：一组模块都访问同一全局简单变量而不是同一全局数据结构，而且不是通过参数表传递该全局变量的信息。

公共耦合：两个模块之间通过一个公共的数据区域传递信息。

内容耦合：一个模块需要涉及到另一个模块的内部信息。

44、测试阶段划分

|  |  |
| --- | --- |
| 测试阶段 | 内容 |
| 单元测试 | 依据详细设计，模块测试，模块功能、性能、接口等 |
| 集成测试 | 依据概要设计，模块间的接口 |
| 系统测试 | 依据需求文档，在真实环境下，验证完整的软件配置项能否和系统正确连接 |
| 确认测试 | 依据需求文档，验证软件与需求的一致性。内部确认测试、Alpha测试、Beta测试、验收测试 |

45、白盒测试（结构测试）

概念：根据内部结构和逻辑来设计测试用例，对程序路径和过程进行测试。主要用于单元测试阶段。

|  |  |  |
| --- | --- | --- |
|  | 定义 | 特点 |
| 语句覆盖 | 被测试程序中的每条语句至少执行一次。 | 对执行逻辑覆盖很低，一般认为是很弱的逻辑覆盖。 |
| 判定覆盖  （分支  覆盖） | 被测程序每个判定表达式至少获得一次“真”值和“假”值（或者程序中每一个判定取“真”分支和取“假”分支至少通过一次。） | 判定覆盖比语句覆盖  更强一些。  判定可以是1个条件，也可以是多个条件的组合。 |
| 条件覆盖 | 每一个判定语句中每个逻辑条件的各种可能的值至少满足一次。 | 条件覆盖和判断覆盖没有包含关系。 |
| 判断/条件覆盖 | 判定中每个条件的所有可能取值（真/假）至少出现一次，并使每个判定本身的判定结果（真/假）也至少出现一次。 | 同时满足判定覆盖  和条件覆盖 |
| 条件组合  覆盖 | 每个判定中的各种可能值的组合都至少出现一次。 | 同时满足判定覆盖、条件覆盖、判定/条件覆盖。 |
| 路径覆盖 | 覆盖被测试程序中所有可能的路径。 |  |
| 基本路径  测试 | 每一条独立路径都执行过（即程序中可执行语句至少执行一次）。 | 测试用例个数与环路复杂度一致。判定为关键控制结点，必须出现在基本路径中。 |
| 循环覆盖 | 循环中每个条件都得到验证。 | 注意数组参数可循环验证 |

46、黑盒测试（功能测试）

概念：黑盒测试基于产品功能规格说明书，从用户角度针对产品特定的功能和特性来进行验证活动，确认每个功能是否得到完整实现，用户能否正常使用这些功能。主要用于集成测试、确认测试和系统测试阶段。

黑盒测试在不知道系统或组件内部结构的情况下进行，不考虑内部逻辑结构，着眼于程序外部结构，在软件接口处进行测试。

试图发现的错误：功能不正确或遗漏；界面错误；数据库访问错误；性能错误；初始化和终止错误等

方法：等价类划分法；边界值分析法；因果图法；判定表驱动法；正交试验设计法；错误推测法 ；功能图法。

|  |
| --- |
| 等价类划分：不同等价类，揭示不同问题；有效等价类/无效等价类。 |
| 边界值分析：1<=x<=10 ，可取x的值为0、1、10和11作为测试数据 |
| 错误推测：依靠测试人员的经验和直觉。 |
| 判定表：最适合描述在多个逻辑条件取值的组合所构成的复杂情况下，分别要执行哪些不同的动作。 |
| 因果图：根据输入条件与输出结果之间的因果关系来设计测试用例。 |

47、软件过程改进CMMI

|  |  |  |
| --- | --- | --- |
| 成熟度等级 | 过程域 | 关键特征 |
| 优化级【L5】 | 组织级改革与实施、因果分析和解决方案 | 持续优化 |
| 定量管理级【L4】 | 组织级过程性能、定量项目管理 | 量化式管理 |
| 已定义级【L3】 | 需求开发、技术解决方案、产品集成、验证、确认、组织级过程焦点、组织级过程定义、组织级培训、集成项目管理、风险管理、集成化的团队、决策分析和解决方案、组织级集成环境 | 组织级，文档化标准化 |
| 已管理级【L2】 | 需求管理、项目计划、配置管理、项目监督与控制、供应商合同管理、度量和分析、过程和产品质量保证 | 项目级可重复 |
| 混乱级【L1】 |  |  |

48、嵌入式微处理器体系结构

|  |  |  |  |
| --- | --- | --- | --- |
|  | 定义 | 特点 | 典型应用 |
| 冯·诺依曼结构 | 冯·诺依曼结构也称普林斯顿结构，是一种将程序指令存储器和数据存储器合并在一起的存储器结构。 | 指令与数据存储器合并在一起。  指令与数据都通过相同的数据总线传输。 | 一般用于PC处理器，如I3、I5、I7处理器。  注：常规计算机属于冯·诺依曼结构 |
| 哈佛结构 | 哈佛结构是一种并行体系结构，它的主要特点是将程序和数据存储在不同的存储空间中，即程序存储器和数据存储器是两个独立的存储器，每个存储器独立编址、独立访问。 | 指令与数据分开存储，可以并行读取，有较高数据的吞吐率。  有4条总线：指令和数据的数据总线与地址总线。 | 一般用于嵌入式系统处理器。  注：DSP属于哈佛结构 |

49、MVC

Model（模型）是应用程序中用于处理应用程序数据逻辑的部分。通常模型对象负责在数据库中存取数据。

View（视图）是应用程序中处理数据显示的部分。通常视图是依据模型数据创建的。

Controller（控制器）是应用程序中处理用户交互的部分。通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据。

J2EE体系结构中：

视图（View）：JSP

控制（Controller）：Servlet

模型（Model）：Entity Bean、Session Bean

50、性能指标

（1）、主频和CPU时钟周期（Clock Cycle） ：主频又称为时钟频率，时钟周期是时钟频率的倒数。如主频为1GHz，则说明1秒钟有1G个时钟周期，每个时钟周期为1000\*1000\*1000/1G=1ns。

主频 = 外频 \* 倍频

（2）、指令周期（Instruction Cycle）：取出并执行一条指令的时间。

（3）、总线周期（BUS Cycle）：也就是CPU完成一次访问MEM或I/O端口操作所用的时间。

（4）、指令周期、总线周期和时钟周期之间的关系：一个指令周期由若干个总线周期组成，而一个总线周期时间又包含有若干个时钟周期（也可说一个指令周期包含若干个时钟周期）。

（5）、MIPS：每秒处理的百万级的机器语言指令数，主要用于衡量标量机的性能。

（6）、MFLOPS：每秒百万个浮点操作，不能反映整体情况，只能反映浮点运算情况，主要用于衡量t向量机性能。

51、数据库故障与恢复

|  |  |  |
| --- | --- | --- |
| 故障关系 | 故障原因 | 解决方法 |
| 事务本身的可预期故障 | 本身逻辑 | 在程序中预先设置Rollback语句 |
| 事务本身的不可预期故障 | 算术溢出、违反存储保护 | 由DBMS的恢复子系统通过日志，撤消事务对数据库的修改，回退到事务初始状态 |
| 系统故障 | 系统停止运转 | 通常使用检查点法 |
| 介质故障 | 外存被破坏 | 一般使用日志重做业务 |

52、总线：

总线的基本概念：总线是一组能为多个部件分时共享的信息传送线，用来连接多个部件并为之提供信息交换通路。

特点：

挂接在总线上的多个部件只能分时向总线发送数据，但可同时从总线接收数据。

通过总线复用方式可以减少总线中信号线的数量，以较少的信号线传输更多的信息。

总线分类：

（1）从功能上来对总线进行划分：数据总线、地址总线和控制总线

（2）从数据传输的方式划分为并行总线和串行总线

串行总线：数据是一位一位地进行传输的，在传输中每一位数据都占据一个固定的时间长度。常见串行总线：RS232、SPI、I2C、USB、CAN、IEEE 1394等。【长距离，传输波特率可调整，正确性依赖于校验码，数据传输方式可以使用多种】

并行总线：将数据字节的各位用多条数据线同时进行传送。常见并行总线ISA、PCI、VME等。【短距离】

53、WEB设计维度：

（1）从架构来看：MVC，MVP，MVVM，REST，Webservice，微服务。

（2）从缓存来看：MemCache，Redis，Squid。

（3）从并发分流来看：集群（负载均衡）、CDN。

（4）从数据库来看：主从库（主从复制），内存数据库，反规范化技术，NoSQL，分区（分表）技术，视图与物化视图。

（5）从持久化来看：Hibernate，Mybatis。

（6）从分布存储来看：Hadoop，FastDFS，区块链。

（7）从数据编码看： XML，JSON。

（8）从Web应用服务器来看： Apache，WebSphere，WebLogic，Tomcat，JBOSS，IIS。

（9）其它：静态化，有状态与无状态，响应式Web设计。

54、集群：（1）应用服务器集群；（2）主从集群。

55、负载均衡技术：（1）应用层负载均衡：http重定向、反向代理服务器；（2）传输层负载均衡：DNS域名解析负载均衡、基于NAT的负载均衡；（3）硬件负载均衡：F5；（6）软件负载均衡：LVS、Nginx、HAproxy。

56、负载均衡算法

静态算法（不考虑动态负载）：

轮转算法：轮流将服务请求（任务）调度给不同的节点（即：服务器）。

加权轮转算法：考虑不同节点处理能力的差异。

源地址哈希散列算法：根据请求的源IP地址，作为散列键从静态分配的散列表找出对应的节点。

目标地址哈希散列算法：根据请求目标IP做散列找出对应节点。

随机算法：随机分配，简单，但不可控。

动态算法（考虑动态负载）

最小连接数算法：每个节点处理能力相同的情况下，新请求分配给当前活动请求数量最少的节点。

加权最小连接数算法：考虑节点处理能力不同，按最小连接数分配。

加权百分比算法：考虑了节点的利用率、硬盘速率、进程个数等，使用利用率来表现剩余处理能力。

57、Redis 与 Memcache 能力比较

|  |  |  |
| --- | --- | --- |
| 工作 | MemCache | Redis |
| 数据类型 | 简单key/value结构 | 丰富的数据结构 |
| 持久性 | 不支持 | 支持 |
| 分布式存储 | 客户端哈希分片/一致性哈希 | 多种方式，主从、Sentinel、Cluster等 |
| 多线程支持 | 支持 | 不支持（Redis6.0开始支持） |
| 内存管理 | 私有内存池/内存池 | 无 |
| 事务支持 | 不支持 | 有限支持 |
| 数据容灾 | 不支持，不能做数据恢复 | 支持，可以在灾难发生时，恢复数据 |

58、Redis集群切片的常见方式

|  |  |
| --- | --- |
| 集群切片方式 | 核心特点 |
| 客户端分片 | 在客户端通过key的hash值对应到不同的服务器。 |
| 中间件实现分片 | 在应用软件和Redis中间，例如：Twemproxy、Codis等，由中间件实现服务到后台Redis节点的路由分派。 |
| 客户端服务端协作分片 | 客户端与服务端协作完成分片处理。 |

59、Redis分布式存储方案

|  |  |
| --- | --- |
| 分布式存储方案 | 核心特点 |
| 主从（Master/Slave）模式 | 一主多从，故障时手动切换。 |
| 哨兵（Sentinel）模式 | 有哨兵的一主多从，主节点故障自动选择新的主节点。 |
| 集群（Cluster）模式 | 分节点对等集群，分slots，不同slots的信息存储到不同节点。 |

60、Redis数据分片方案

|  |  |  |
| --- | --- | --- |
| 分片方案 | 分片方式 | 说明 |
| 范围分片 | 按数据范围值来做分片 | 例：按用户编号分片，0-999999映射到实例A；1000000-1999999映射到实例B。 |
| 哈希分片 | 通过对key进行hash运算分片 | 可以把数据分配到不同实例，这类似于取余操作，余数相同的，放在一个实例上。 |
| 一致性哈希分片 | 哈希分片的改进 | 可以有效解决重新分配节点带来的无法命中问题。 |

61、缓存与数据库的协作

数据读取：根据key从缓存读取；若缓存中没有，则根据key在数据库中查找；读取到“值”之后，更新缓存。

数据写入：根据key值写数据库；根据key更新缓存。

62、REST概念：REST（Representational State Transfer，表述性状态转移）是一种只使用HTTP和XML进行基于Web通信的技术，可以降低开发的复杂性，提高系统的可伸缩性。

REST的五个原则：网络上的所有事物都被抽象为资源；每个资源对应一个唯一的资源标识；通过通用的连接件接口对资源进行操作；对资源的各种操作不会改变资源标识；所有的操作都是无状态的。

63、响应式Web设计：响应式WEB设计是一种网络页面设计布局，其理念是：集中创建页面的图片排版大小，可以智能地根据用户行为以及使用的设备环境进行相对应的布局。方法：采用流式布局和弹性化设计、响应式图片。

64、主从数据库结构特点：

一般：一主多从，也可以多主多从。

从库做写操作，从库做读操作。

主从复制步骤：

主库（Master）更新数据完成前，将操作写binlog日志文件。

从库（Salve）打开I/O线程与主库连接，做binlog dump process，并将事件写入中继日志。

从库执行中继日志事件，保持与主库一致。

65、反规范化的技术手段以及优缺点

|  |  |
| --- | --- |
| 技术手段 | 说明 |
| 增加派生性冗余列 | 已有单价和数量列，增加“总价”列 |
| 增加冗余列 | 已有学号列，增加“姓名”列 |
| 重新组表 | 把拆分的表重新组表 |
| 分割表 | 把用户表做水平分割，长沙的用户存在长沙，上海的用户存在上海 |

反规范化的优点：连接操作少，检索快、统计快；需要查的表减少，检索容易。

|  |  |
| --- | --- |
| 反规范化的缺点 | 解决方案 |
| 数据冗余，需要更大存储空间 | 无解 |
| 插入、更新、删除操作开销更大 | 无解 |
| 数据不一致  可能产生添加、修改、删除异常 | 1、触发器数据同步  2、应用程序数据同步  3、物化视图 |
| 更新和插入代码更难写 | 无解 |

66、视图的优点：（1）视图能简化用户的操作（2）视图机制可以使用户以不同的方式查询同一数据（3）视图对数据库重构提供了一定程度的逻辑独立性（4）视图可以对机密的数据提供安全保护其中物化视图：将视图的内容物理存储起来，其数据随原始表变化，同步更新。

67、分表和分区

|  |  |  |
| --- | --- | --- |
|  | 分区 | 分表 |
| 共性 | 1、都针对数据表 2、都使用了分布式存储  3、都提升了查询效率 4、都低数据库的频繁I/O压力值 | |
| 差异 | 逻辑上还是一张表 | 逻辑上已是多张表。 |

68、分区的优点：（1）相对于单个文件系统或是硬盘，分区可以存储更多的数据。（2）数据管理比较方便，比如要清理或废弃某年的数据，就可以直接删除该日期的分区数据即可。（3）精准定位分区查询数据，不需要全表扫描查询，大大提高数据检索效率。（4）可跨多个分区磁盘查询，来提高查询的吞吐量。（5）在涉及聚合函数查询时，可以很容易进行数据的合并。

69、关系型数据库和NoSQL对比

|  |  |  |
| --- | --- | --- |
| 对比维度 | 关系数据库 | NoSQL |
| 应用领域 | 面向通用领域 | 特定应用领域 |
| 数据容量 | 有限数据 | 海量数据 |
| 数据类型 | 结构化数据【二维表】 | 非结构化数据 |
| 并发支持 | 支持并发、但性能低 | 高并发 |
| 事务支持 | 高事务性 | 弱事务性 |
| 扩展方式 | 向上扩展 | 向外扩展 |

70、嵌入式微处理器分类

（1）嵌入式微控制器（MCU：Micro Controller Unit）：又称为单片机，片上外设资源一般比较丰富，适合于控制。

（2）嵌入式微处理器（EMPU：Embedded Micro Processing Unit）： 又称为单板机， 由通用计算机中的CPU发展而来，仅保留和嵌入式应用紧密相关的功能硬件。

（3）嵌入式DSP处理器（DSP：Digital Signal Processor）：专门用于信号处理方面的处理器。

（4）嵌入式片上系统（SOC）：追求产品系统最大包容的集成器件。

成功实现了软硬件的无缝结合，直接在微处理器片内嵌入操作系统的代码模块。

减小了系统的体积和功耗、提高了可靠性和设计生产效率。

71、常见的摘要算法：MD5(128位)，SHA(160位)

72、常见对称密钥加密算法

DES：替换+移位、56位密钥、64位数据块、速度快、密钥易产生

3DES(三重DES)：两个56位的密钥K1、K2

加密：K1加密->K2解密->K1加密

解密：K1解密->K2加密->K1解密 RC-5

IDEA：128位密钥、64位数据块、比DES的加密性好、对计算机功能要求相对低，PGP。

RC-5算法：RSA数据安全公司的很多产品都使用了RC-5。

AES算法：高级加密标准，又称Rijndael加密法，是美国政府采用的一种区块加密标准。

73、常见非对称密钥加密算法

RSA：2048位（或1024位）密钥、计算量极大、难破解

ECC-椭圆曲线算法

Elgamal：安全性依赖于计算有限域上离散对数这一难题。

74、被动攻击

被动攻击：收集信息为主，破坏保密性。

|  |  |  |
| --- | --- | --- |
| 攻击类型 | 攻击名称 | 描述 |
| 被动攻击 | 窃听（网络监听） | 用各种可能的合法或非法的手段窃取系统中的信息资源和敏感信息。 |
| 业务流分析 | 通过对系统进行长期监听，利用统计分析方法对诸如通信频度、通信的信息流向、通信总量的变化等参数进行研究，从而发现有价值的信息和规律。 |
| 非法登录 | 有些资料将这种方式归为被动攻击方式。 |

75、主动攻击

主动攻击的类别主要有：中断（破坏可用性），篡改（破坏完整性），伪造（破坏真实性）

|  |  |  |
| --- | --- | --- |
| 攻击类型 | 攻击名称 | 描述 |
| 主动攻击 | 假冒身份 | 通过欺骗通信系统（或用户）达到非法用户冒充成为合法用户，或者特权小的用户冒充成为特权大的用户的目的。黑客大多是采用假冒进行攻击。 |
| 抵赖 | 这是一种来自用户的攻击，比如：否认自己曾经发布过的某条消息、伪造一份对方来信等。 |
| 旁路控制  【旁路攻击】 | 密码学中是指绕过对加密算法的繁琐分析，利用密码算法的硬件实现的运算中泄露的信息。如执行时间、功耗、电磁辐射等，结合统计理论快速的破解密码系统。 |
| 重放攻击 | 所截获的某次合法的通信数据拷贝，出于非法的目的而被重新发送。  加时间戳能识别并应对重放攻击。 |
| 拒绝服务（DOS） | 对信息或其它资源的合法访问被无条件地阻止。 |
| XSS跨站脚本攻击 | 通过利用网页【开发时留下的漏洞】，通过巧妙的方法注入恶意指令代码到网页 |
| CSRF跨站请求伪造攻击 | 攻击者通过一些技术手段欺骗用户的浏览器与访问一个自己曾经认证过的网站并执行一些操作（如转账或购买商品等）。 |
| 缓冲区溢出攻击 | 利用【缓冲区溢出漏洞】所进行的攻击。在各种操作系统、应用软件中广泛存在。 |
| SQL注入攻击 | 攻击者把SQL命令插入到Web表单，欺骗服务器执行恶意的SQL命令。  SQL注入攻击的方式：【恶意拼接查询】、【利用注释执行非法命令】、【传入非法参数】、【添加额外条件】  抵御SQL攻击的方式包括：【使用正则表达式】、【使用参数化的过滤性语句】、【检查用户输入的合法性】、【用户相关数据加密处理】、【存储过程来执行查询】  【使用专业的漏洞扫描工具】 |

76、安全保护等级

|  |  |  |  |
| --- | --- | --- | --- |
|  | 公民、法人和其它组织权益 | 社会秩序和公共利益 | 国家安全 |
| 用户自主保护级 | 损害 |  |  |
| 系统审计保护级 | 严重损害 | 损害 |  |
| 安全标记保护级 |  | 严重损害 | 损害 |
| 结构化保护级 |  |  | 严重损害 |
| 访问验证保护级 |  |  | 特别严重损害 |

77、恢复块方法与N版本程序设计对比

|  |  |  |
| --- | --- | --- |
|  | 恢复块方法 | N版本程序设计 |
| 硬件运行环境 | 单机 | 多机 |
| 错误检测方法 | 验证测试程序 | 表决 |
| 恢复策略 | 后向恢复 | 前向恢复 |
| 实时性 | 差 | 好 |

78、Redis持久化

RDB：传统数据库中快照的思想。指定时间间隔将数据进行快照存储。

AOF：传统数据库中日志的思想，把每条改变数据集的命令追加到AOF文件末尾，这样出问题了，可以重新执行AOF文件中的命令来重建数据集。

|  |  |  |
| --- | --- | --- |
| 对比维度 | RDB持久化 | AOF持久化 |
| 备份量 | 重量级的全量备份，保存整个数据库 | 轻量级增量备份，一次只保存一个修改命令 |
| 保存间隔时间 | 保存间隔时间长 | 保存间隔时间短，默认1秒 |
| 还原速度 | 数据还原速度快 | 数据还原速度慢 |
| 阻塞情况 | save会阻塞，但bgsave或者自动不会阻塞 | 无论是平时还是AOF重写，都不会阻塞 |
| 数据体积 | 同等数据体积：小 | 同等数据体积：大 |
| 安全性 | 数据安全性：低，容易丢数据 | 数据安全性：高，根据策略决定 |

79、Session有状态和无状态问题

无状态服务（stateless service）对单次请求的处理，不依赖其他请求，也就是说，处理一次请求所需的全部信息，要么都包含在这个请求里，要么可以从外部获取到（比如说数据库），服务器本身不存储任何信息。

有状态服务（stateful service）则相反，它会在自身保存一些数据，先后的请求是有关联的。

80、性能评估

|  |  |  |
| --- | --- | --- |
| 方法 | 描述 | 特点 |
| 时钟频率法 | 以时钟频率高低衡量速度 | 仅考虑CPU |
| 指令执行速度法 | 表示机器运算速度的单位是MIPS | 仅考虑CPU |
| 等效指令速度法（吉普森混合法） | 通过各类指令在程序中所占的比例（*Wi*）进行计算得到的。 | 仅考虑CPU，综合考虑指令比例不同的问题。 |
| 数据处理速率法（PDR） | PDR值的方法来衡量机器性能，PDR值越大，机器性能越好。 | PDR = L/R  仅考虑CPU+存储 |
| 综合理论性能法（CTP） | CTP用MTOPS表示。CTP的估算方法是，首先算出处理部件每个计算单元的有效计算率，再按不同字长加以调整，得出该计算单元的理论性能，所有组成该处理部件的计算单元的理论性能之和即为CTP。 | 仅考虑CPU+存储 |
| 基准程序法 | 把应用程序中用得最多、最频繁的那部分核心程序作为评估计算机系统性能的标准程序，称为基准测试程序（benchmark）。 | 综合考虑多部分，基准程序法是目前一致承认的测试系统性能的较好方法。 |

81、可靠性指标

平均无故障时间 → (MTTF) MTTF=1/λ，λ为失效率

平均故障修复时间 → (MTTR) MTTR=1/μ，μ为修复率

平均故障间隔时间 → (MTBF) MTBF = MTTR + MTTF

系统可用性 → MTTF/(MTTR+MTTF)×100%

在实际应用中，一般MTTR很小，所以通常认为MTBF≈MTTF。

82、基于架构的软件设计（ABSD）

（ABSD方法是架构驱动，即强调由业务【商业】、质量和功能需求的组合驱动架构设计。

ABSD方法有三个基础。第一个基础是功能的分解。在功能分解中，ABSD方法使用已有的基于模块的内聚和耦合技术；第二个基础是通过选择架构风格来实现质量和业务需求；第三个基础是软件模板的使用。软件模板利用了一些软件系统的结构。

视角与视图：从不同的视角来检查，所以会有不同的视图。

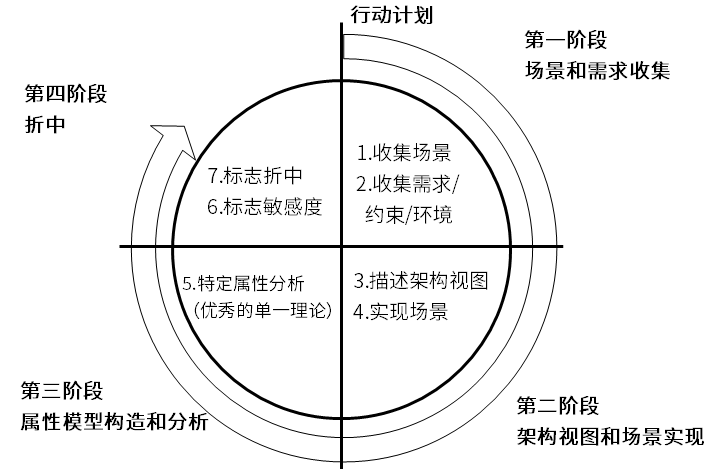
用例用来捕获功能需求、特定场景【刺激、环境、响应】用来捕获质量需求。

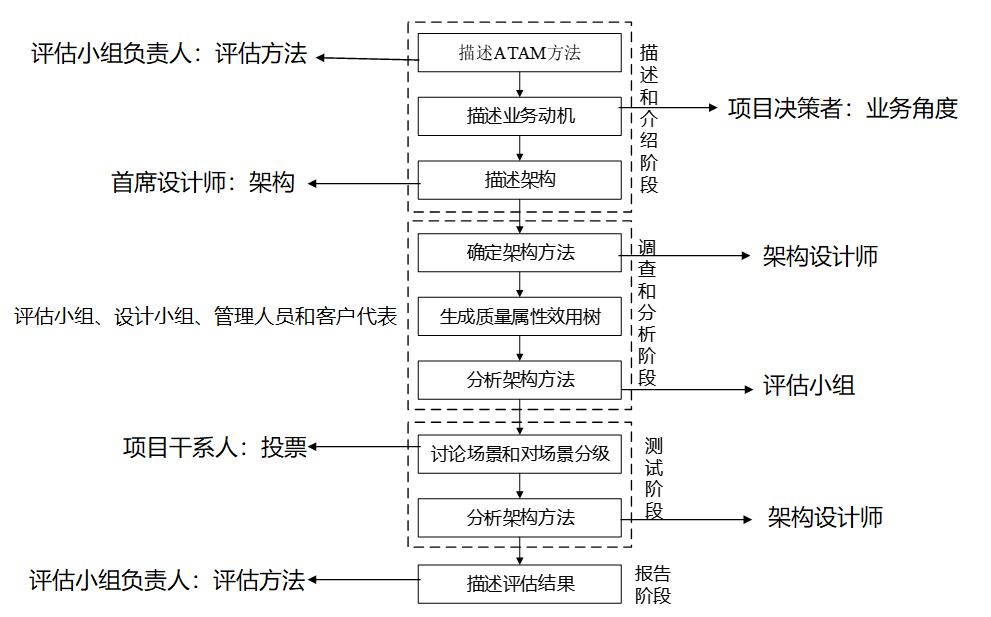
开发过程：



83、架构权衡分析法（ATAM）

（在SAAM的基础上发展起来的，主要针对性能、实用性、安全性和可修改性，在系统开发之前，对这些质量属性进行评价和折中。）





84、风险点与非风险点、敏感点与权衡点

风险点：系统架构风险是指架构设计中潜在的、存在问题的架构决策所带来的隐患。

非风险点：是指不会带来隐患，一般以“XXX要求是可以实现【或接受】的”方式表达。

敏感点：敏感点是一个或多个构件（和/或构件之间的关系）的特性，它能影响系统的某个质量属性。

权衡点：影响多个质量属性的特性，是多个质量属性的敏感点。

85、质量属性

（1）性能

性能（performance）是指系统的响应能力，即要经过多长时间才能对某个事件做出响应，或者在某段时间内系统所能处理的事件的个数。例如：a.同时支持1000并发；b.响应时间小于1s；c.显示分辨率达到4K。

代表参数：响应时间、吞吐量 设计策略：优先级队列、资源调度

（2）可用性

可用性（availability）是系统能够正常运行的时间比例。经常用两次故障之间的时间长度或在出现故障时系统能够恢复正常的速度来表示。例如：a.主服务器故障，1分钟内切换至备用服务器；b.系统故障，1小时内修复；c.系统支持7×24小时工作。

代表参数：故障间隔时间 设计策略：冗余、心跳线

（3）安全性

安全性（security）是指系统在向合法用户提供服务的同时能够阻止非授权用户使用的企图或拒绝服务的能力。安全性又可划分为机密性【信息不泄露给未授权的用户】、完整性【防止信息被篡改】、不可否认性【不可抵赖】及可控性【对信息的传播及内容具有控制的能力】等特性。例如：a.可抵御SQL注入攻击；b.对计算机的操作都有完整记录；c.用户信息数据库授权必须保证99.9%可用。

设计策略：追踪审计

（4）可修改性

可修改性（modifiability）是指能够快速地以较高的性能价格比对系统进行变更的能力。通常以某些具体的变更为基准，通过考察这些变更的代价衡量可修改性。（可扩展性与之相近）例如：a.更改系统报表模块，必须在2人周内完成；b.对Web界面风格进行修改，修改必须在4人月内完成。

主要策略：信息隐藏（二义性：良好的封装能够做到信息隐藏，一般归于可修改性策略；信息隐藏也能够体现在安全性当中）

（5）易用性

易用性关注的是对用户来说完成某个期望任务的容易程度和系统所提供的用户支持的种类。例如：a.界面友好；b.新用户学习使用系统时间不超过2小时。

（6）可测试性

软件可测试性是指通过测试揭示软件缺陷的容易程度。例如：a.提供远程调试接口，支持远程调试。

86、三级模式与两层映射

三级模式：外模式对应视图，模式（也称为概念模式）对应数据库表，内模式对应物理文件。

两层映像：外模式-模式映像，模式-内模式映像；两层映像可以保证数据库中的数据具有较高的逻辑独立性和物理独立性。

87、数据库设计过程



88、三个范式

第一范式（1NF）：在关系模式R中，当且仅当所有域只包含原子值，即每个属性都是不可再分的数据项，则称关系模式R是第一范式。

第二范式（2NF）--消除非主属性对码的部分函数依赖：当且仅当关系模式R是第一范式（1NF），且每一个非主属性完全依赖候选键（没有不完全依赖）时，则称关系模式R是第二范式。

第三范式（3NF）--消除非主属性对码的传递函数依赖：当且仅当关系模式R是第二范式（2NF），且R中没有非主属性传递依赖于候选键时，则称关系模式R是第三范式。

89、NoSQL分类

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 分类 | 典型应用场景 | 数据模型 | 优点 | 缺点 | 举例 |
| 键值  （key-value） | 内容缓存，主要用于处理大量数据的高访问负载，也用于一些日志系统等等。 | Key指向Value的键值对，通常用hash table来实现 | 查找速度快 | 数据无结构化，通常只被当作字符串或者二进制数据 | Redis,Tokyo Cabinet/  Tyrant, Voldemort,  Oracle BDB |
| 列存储数据库 | 分布式的文件系统 | 以列簇式存储，将同一列数据存在一起 | 查找速度快，可扩展性强，更容易进行分布式扩展 | 功能相对局限 | HBase, Cassandra, Riak |
| 文档型数据库 | Web应用（与Key-Value类似，Value是结构化的，不同的是数据库能够了解Value的内容） | Key-Value对应的键值对，Value为结构化数据 | 数据结构要求不严格，表结构可变，不需要像关系型数据库一样需要预先定义表结构 | 查询性能不高，而且缺乏统一的查询语法。 | CouchDB, MongoDb |
| 图形数据库  （Graph） | 社交网络，推荐系统等。专注于构建关系图谱 | 图结构 | 利用图结构相关算法。比如最短路径寻址，N度关系查找等 | 很多时候需要对整个图做计算才能得出需要的信息，而且这种结构不太好做分布式的集群方案。 | Neo4J, InfoGrid,  Infinite Graph |

90、封锁协议

一级封锁协议。事务T在修改数据R之前必须先对其加X锁，直到事务结束才释放。可防止丢失修改。

二级封锁协议。一级封锁协议加上事务T在读取数据R之前先对其加S锁，读完后即可释放S锁。可防止丢失修改，还可防止读“脏”数据。

三级封锁协议。一级封锁协议加上事务T在读取数据R之前先对其加S锁，直到事务结束才释放。可防止丢失修改、防止读“脏”数据与防止数据重复读。

两段锁协议。可串行化的。可能发生死锁。

91、备份

（1）冷备份也称为静态备份，是将数据库正常关闭，在停止状态下，将数据库的文件全部备份（复制）下来。

（2）热备份也称为动态备份，是利用备份软件，在数据库正常运行的状态下，将数据库中的数据文件备份出来。

（3）完全备份：备份所有数据

（4）差量备份：仅备份上一次完全备份之后变化的数据

（5）增量备份：备份上一次备份之后变化的数据

（6）日志文件 ：事务日志是针对数据库改变所做的记录，它可以记录针对数据库的任何操作，并将记录结果保存在独立的文件中。

92、TCP与UDP

|  |  |  |
| --- | --- | --- |
|  | TCP | UDP |
| 共同点 | 基于IP协议的传输层协议，可以端口寻址 | |
| 不同点 | 面向连接（连接管理）、三次握手、流量控制、差错校验和重传、IP数据报按序接收不丢失不重复、可靠性强、牺牲通信量、效率低 | 不可靠、无连接、错误检测功能弱，无拥塞控制、无流量控制，有助于提高传输的高速率性。  不对无序IP数据报重新排序、不负责重传、不消除重复IP数据报、不对已收到的数据报进行确认、不负责建立或终止连接，这些由UDP进行通信的应用程序进行处理。 |
| 相关协议 | HTTP、FTP、Telnet、POP3、SMTP | DNS、DHCP、TFTP、SNMP |

93、WPDRRC模型

WPDRRC模型：WPDRRC信息安全模型是我国“八六三”信息安全专家组提出的适合中国国情的信息系统安全保障体系建设模型。

WPDRRC模型包括6个环节和3大要素。

6个环节包括：预警、保护、检测、响应、恢复和反击。模型蕴涵的网络安全能力主要是预警能力、保护能力、检测能力、响应能力、恢复能力和反击能力。

3大要素包括人员、策略和技术。

94、IPv6是设计用于替代现行版本IP协议（IPv4）的下一代IP协议。

（1）IPv6地址长度为128位，地址空间增大了296倍；

（2）灵活的IP报文头部格式。使用一系列固定格式的扩展头部取代了IPv4中可变长度的选项字段。IPv6中选项部分的出现方式也有所变化，使路由器可以简单路过选项而不做任何处理，加快了报文处理速度；

（3）IPv6简化了报文头部格式，字段只有8个，加快报文转发，提高了吞吐量；

（4）提高安全性。身份认证和隐私权是IPv6的关键特性；

（5）支持更多的服务类型；

（6）允许协议继续演变，增加新的功能，使之适应未来技术的发展；

95、区块链技术

（1）【区块链】 ≠ 比特币，比特币底层采用了区块链技术。比特币交易在我国定性为【非法运用】。

（2）区块链的特点：

去中心化：由于使用分布式核算和存储，不存在中心化的硬件或管理机构，任意节点的权利和义务都是均等的，系统中的数据块由整个系统中具有维护功能的节点来共同维护。

开放性：系统是开放的，如：区块链上的【交易信息是公开的】，不过【账户身份信息是高度加密的】。

自治性：区块链采用基于协商一致的规范和协议（比如一套公开透明的算法）使得整个系统中的所有节点能够在去信任的环境自由安全的交换数据，使得对“人”的信任改成了对机器的信任，任何人为的干预不起作用。

安全性（信息不可篡改）：数据在多个节点存储了多份，篡改数据得改掉51%节点的数据，这太难。同时，还有其它安全机制，如：比特币的每笔交易，都由付款人用私钥签名，证明确实是他同意向某人付款，其它人无法伪造。

匿名性（去信任）：由于节点之间的交换遵循固定的算法，其数据交互是无需信任的（区块链中的程序规则会自行判断活动是否有效），因此交易对手无须通过公开身份的方式让对方自己产生信任，对信用的累积非常有帮助。

96、数学建模是一种数学的思考方法，是运用数学的语言和方法，通过抽象和简化，建立能近似刻画并解决实际问题的模型的一种强有力的数学手段。

97、侵权判断

中国公民、法人或者其他组织的作品，不论是否发表，都享有著作权。

开发软件所用的思想、处理过程、操作方法或者数学概念不受保护

著作权法不适用于下列情形：

法律、法规，国家机关的决议、决定、命令和其他具有立法、行政、司法性质的文件，及其官方正式译文；

时事新闻；

历法、通用数表、通用表格和公式。

|  |  |
| --- | --- |
| 不侵权 | 侵权 |
| * 个人学习、研究或者欣赏； * 适当引用； * 公开演讲内容 * 用于教学或科学研究 * 复制馆藏作品； * 免费表演他人作品； * 室外公共场所艺术品临摹、绘画、摄影、录像； * 将汉语作品译成少数民族语言作品或盲文出版。 | * 未经许可，发表他人作品； * 未经合作作者许可，将与他人合作创作的作品当作自己单独创作的作品发表的； * 未参加创作，在他人作品署名； * 歪曲、篡改他人作品的； * 剽窃他人作品的； * 使用他人作品，未付报酬； * 未经出版者许可，使用其出版的图书、期刊的版式设计的。 |

98、保护对象和范围

|  |  |  |
| --- | --- | --- |
| 法律法规名称 | 保护对象及范围 | 注意事项 |
| 著作权法 | 著作权  文学、绘画、摄影等作品 | 1、不需要申请，作品完成即开始保护  2、绘画或摄影作品原件出售（赠予）著作权还归原作者，原件拥有者有：所有权、展览权。 |
| 软件著作权法  计算机软件保护条例 | 软件著作权  软件作品 | 1、不需要申请，作品完成即开始保护  2、登记制度便于举证 |
| 专利法 | 专利权 | 需要申请，专利权有效期是从申请日开始计算 |
| 商标法 | 商标权 | 需要申请，核准之日起商标受保护 |
| 反不正当竞争法 | 商业秘密权 | 1、商业秘密包括技术与经营两个方面  2、必须有保密措施才能认定商业秘密 |

99、保护期限

|  |  |  |
| --- | --- | --- |
| 客体类型 | 权力类型 | 保护期限 |
| 公民作品 | 署名权、修改权、保护作品完整权 | 没有限制 |
| 发表权、使用权和获得报酬权 | 作者终生及其死亡后的50年（第50年的12月31日） |
| 单位作品 | 发表权、使用权和获得报酬权 | 50年（首次发表后的第50年的12月31日） |
| 公民软件产品 | 署名权、修改权 | 没有限制 |
| 发表权、复制权、发行权、出租权、信息网络传播权、翻译权、使用许可权、获得报酬权、转让权 | 作者终生及死后50年（第50年12月31日）。合作开发，以最后死亡作者为准。 |
| 单位软件产品 | 发表权、复制权、发行权、出租权、信息网络传播权、翻译权、使用许可权、获得报酬权、转让权 | 50年（首次发表后的第50年的12月31日），若期间未发表，不保护 |
| 注册商标 | | 有效期10年（若注册人死亡或倒闭1年后，未转移则可注销，期满后6个月内必须续注） |
| 发明专利权 | | 保护期为20年（从申请日开始） |
| 实用新型和外观设计专利权 | | 保护期为10年（从申请日开始） |
| 商业秘密 | | 不确定，公开后公众可用 |

100、知识产权人确定

|  |  |  |  |
| --- | --- | --- | --- |
| 情况说明 | | 判断说明 | 归属 |
| 作品 | 职务作品 | 利用单位的物质技术条件进行创作，并由单位承担责任的 | 除署名权外其他著作权归单位 |
| 有合同约定，其著作权属于单位 | 除署名权外其他著作权归单位 |
| 其他 | 作者拥有著作权，单位有权在业务范围内优先使用 |
| 软件 | 职务  作品 | 属于本职工作中明确规定的开发目标 | 单位享有著作权 |
| 属于从事本职工作活动的结果 | 单位享有著作权 |
| 使用了单位资金、专用设备、未公开的信息等物质、技术条件，并由单位或组织承担责任的软件 | 单位享有著作权 |
| 专利权 | 职务  作品 | 本职工作中作出的发明创造 | 单位享有专利 |
| 履行本单位交付的本职工作之外的任务所作出的发明创造 | 单位享有专利 |
| 离职、退休或调动工作后1年内，与原单位工作相关 | 单位享有专利 |

|  |  |  |  |
| --- | --- | --- | --- |
| 情况说明 | | 判断说明 | 归属 |
| 作品  软件 | 委托  创作 | 有合同约定，著作权归委托方 | 委托方 |
| 合同中未约定著作权归属 | 创作方 |
| 合作  开发 | 只进行组织、提供咨询意见、物质条件或者进行其他辅助工作 | 不享有著作权 |
| 共同创作的 | 共同享有，按人头比例。  成果可分割的，可分开申请。 |
| 商标 | | 谁先申请谁拥有（除知名商标的非法抢注）  同时申请，则根据谁先使用（需提供证据）  无法提供证据，协商归属，无效时使用抽签（但不可不确定） | |
| 专利 | | 谁先申请谁拥有  同时申请则协商归属，协商不成则同时驳回双方的专利申请 | |

更多备考资料和学习福利，可扫码添加希赛嘉儿老师，申请入群

